Augmented Learning Modes for Internet Routing

John McCaffery
School of Computer Science
University of St Andrews
St Andrews, UK

Alan Miller
School of Computer Science
University of St Andrews
St Andrews, UK

Iain Oliver
School of Computer Science
University of St Andrews
St Andrews, UK

Colin Allison
School of Computer Science
University of St Andrews
St Andrews, UK

***Abstract*— As the Internet continues to establish itself as a utility, like power, transport or water, it becomes increasingly important to provide an engaging educational experience about its operation for students in related STEM disciplines such as Computer Science and Electrical Engineering. Routing is a core functionality of the global Internet. It can be used as an example of where theory meets practice, where algorithms meet protocols and where science meets engineering. Routing protocols can be included in the Computer Science curriculum in distributed systems, computer networking, algorithms, data structures, and graph theory. While there is a plethora of computer networking textbooks, and copious information of varying quality about the Internet spread across the Web, there is still an essential need for exploratory learning facilities of the type that support group work, experimentation and experiential learning. This paper reports on work using open virtual worlds to provide a multi-user interactive learning environment for Internet routing which exemplifies the capabilities of emerging immersive education technologies to augment conventional practice. The functionality of the learning environment is illustrated through examples and the underlying system which was built to support the routing simulations is explained.**

Keywords—Open Virtual Worlds, Internet Routing, Computer Networking Education

I. INTRODUCTION

Internet routing algorithms and protocols are a type of learning topic that can be used in multiple STEM education contexts. By reference to the Internet they can be used as an example of where theory meets practice, where algorithms meet protocols and where science meets engineering. They can be included in the curriculum in modules on distributed systems, computer networking, algorithms and data structures, and graph theory. They can be tailored to fit almost any level in higher education from First year to Masters. There are many good educational texts on computer networking such as [1-3] which include sections on Internet routing protocols illustrated by relatively simple examples, obviously constrained by the nature of the medium. Well-crafted online courses by networking experts such as the "Introduction to Computer Networking" MOOC [4] offer a media-rich, self-learn, free alternative to the traditional text book through the use of videos, quizzes, animations and forums. Although a welcome addition to the networking education landscape, like most MOOCs it is open to the criticism that it relies solely on a behaviorist pedagogy where information is transferred from teacher to student rather than stimulating critical, creative and original thinking skills in the learner [5].

With respect to textbooks and online courses there remains a strong need for complementary learning resources that support interaction and experimentation in order to allow for knowledge formation in addition to information assimilation.

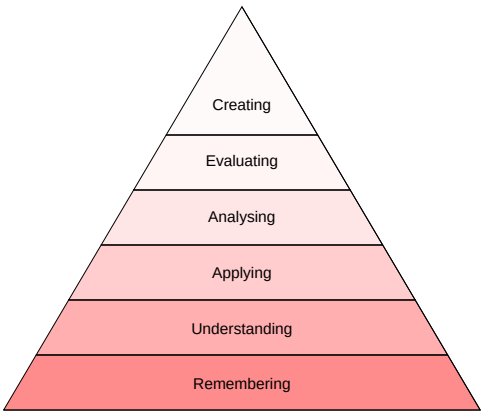


Figure 1: Anderson's Revision of Bloom's Levels of Cognitive Learning

In terms of Anderson's revision [6] of Blooms taxonomy of educational objectives (see Fig. 1) there is a need to support the higher levels of learning achievement - Apply, Analyse, Evaluate, Create - as well as Remember, Understand - and to avoid going down either a solely theoretical or solely practical path. A version of the taxonomy adapted for Computer Science [7] by Fuller identifies the semi-independence of *interpreting* and *producing* with respect to computer programming. If this distinction is carried over to computer networks we note that there is often little opportunity in the crowded curriculum to *produce* and the emphasis is heavily on *interpreting*; hence our motivation for augmenting conventional instructional materials with exploratory learning environments.

The paper proceeds as follows: section 2 gives an overview of open virtual worlds (OVW) and their use in education; section 3 describes routing algorithms used widely in the Internet and typically included in the

curriculum; section 4 shows how OVW can be used to augment learning about the Internet's routing protocols; sections 5 and 6 explain how the system was designed and implemented, and how users interact with it; section 7 concludes.

II. Open Virtual Worlds

Virtual Worlds are a novel type of Internet application where users interact with an immersive 3D environment and with each other through avatars. Part of the motivation for using Virtual Worlds is that students readily engage with them, often more so than with conventional learning materials and contexts [8, 9]. Virtual Worlds for learning have been created to support topics including WiFi experimentation [10], electro-magnetic theory [11], programming algorithms [9, 12], HCI [13], archaeology [14], space science [15], cultural heritage [16], humanitarian aid [17] and linguistic and cultural competence for American forces [18].

Second Life [19] was pioneering in its global reach but it was not designed for education and several commentators have highlighted problem areas that arise when using it for that purpose [20, 21]. These include: commercial cost, code size restrictions, lack of an integrated development environment, difficulty of coursework marking due to the ownership and permissions system, poor quality of experience due to remote servers, firewall blocking by campus computing services, a lack of facilities for copying and sharing content and backing up work outside of the virtual world. Other issues arise – those of ethics, trust, privacy – and the technological barriers to running a class exercise with software which does not scale. In recent years OpenSim [22] has increasingly displaced Second Life (SL) as the platform of choice for developing immersive learning environments. OpenSim is an open source project which uses the same protocols as Second Life so is compatible with any SL compatible viewer/client including the SL viewer itself, Hippo [23], Meerkat [24] and Phoenix [25]. This software compatibility has resulted in OpenSim becoming a de facto standard for Virtual Worlds; as it is freely available we use the term Open Virtual Worlds (OVW).

While OpenSim offers solutions to many of the significant drawbacks encountered with Second Life there are still features (or the lack thereof) inherited from Second Life which act as barriers to exploiting the potential of Virtual Worlds in educational settings - programmability in particular - and section 4 shows how this constraint has been overcome in order to create a sophisticated interactive routing simulation environment.

III. Internet Routing Algorithms

The Internet is organised into Autonomous Systems (AS). There are two general types of routing protocol: exterior, which guide traffic between AS – notably the Border Gateway Protocol BGP [26] - and interior, which manage traffic within a single AS. For example the UK's Joint Academic Network, ja.net, is a single AS. In addition, AS can be organised further into *areas* e.g. ja.net is organised into Metropolitan Area Networks, Regional Networks, Campus Networks, etc.

The two most common types of interior routing protocols in use are Distance Vector (DV) and Link State. RIP (Routing Information Protocol) is a widely deployed DV protocol which was documented originally in 27 pages in RFC 1058 [27] and later in 37 pages in RFC 2453 [28]. OSPF (Open Shortest Path First) is a widely used link state protocol, documented in 224 pages in RFC 2328 [29]. OSPF includes a version of Dijkstra's shortest path algorithm [30], a common component of algorithms and data structure modules, and one that exemplifies the use of Computer Science algorithms in Internet engineering.

There are many online resources for learning about RIP and OSPF, including Wikipedia e.g. [31] and YouTube e.g. [32]. The former is intended for reference rather than educational learning per se, whereas the latter is a type of talking textbook i.e. the videos are didactic, enhancing the information transfer pedagogy with animations, but lacking in interactivity or experimental capabilities. A web-based OSPF tutorial [33] provides user-controlled annotated animations, providing some degree of interactivity. A Java applet [34] allows the learner to specify their own graph (network) and then run the algorithm but suffers from a poor user interface and does not contextualise the algorithm within network engineering and the Internet in general. Routing Archipelago, described in the next section, is an OVW based interactive learning environment which supports various learning modes at various different levels, accommodating video lectures, animations, documents and web pages as well as shared and private simulators for exploring the operation of DV and OSPF.

IV. Routing Archipelago

With the move from Second Life to locally hosted OpenSim virtual worlds, barriers of size, space and cost, have been removed. Whereas an "Island" in Second Life was prohibitively expensive yet still restrictive in terms of content, complexity and size of code, and functionality, it is possible to combine several OpenSim Islands into a single mega-region, accommodating selected content drawn from a wide range of sources.

Routing Archipelago combines videos, reference materials and academic papers with canned simulations, shared simulation areas and private simulation areas, all within a multi-user environment, which a cohort of students can visit together, with or without an instructor, or independently for private study. The main island - "Routing Centre" supports didactic learning. It contains a document repository, two lecture theatres – one for OSPF and one for RIP – and a museum about the history of the Internet. Many smaller islands contain either single user sandbox areas, canned simulations for demonstration purposes, and shared

simulation areas where multiple learners can collaboratively build a network.

This range of learning resources is designed to support all levels of learning – Remember, Understand, Apply, Analyse, Evaluate, Create – discerned by Anderson [6] (see Fig.1) in the context of Internet routing.

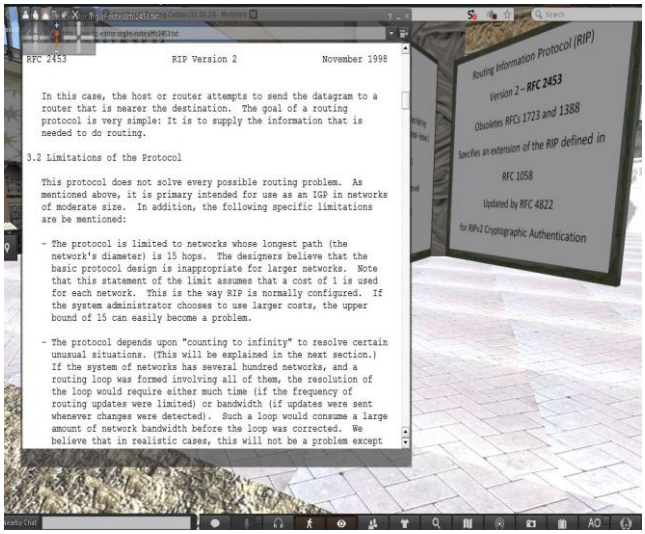


Figure 2: A scene from the Document Repository

Figure 2 shows a scene from the document repository. Large notecards can be touched to load reference material from the web such as RFCs.

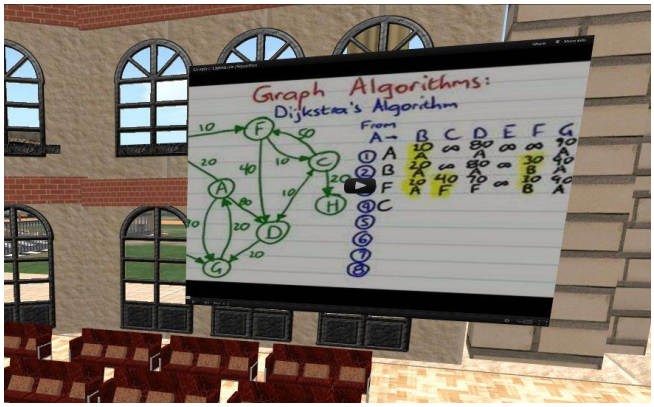
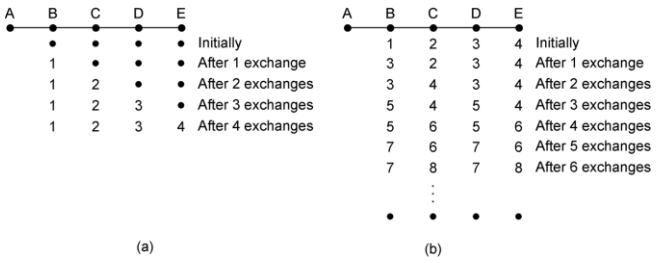


Figure 3: A highly rated YouTube video is streamed www.youtube.com/watch?v=8Ls1RqHCOPw

Figure 3 shows a highly rated education video from YouTube describing Dijkstra's shortest path algorithm being played within the lecture theatre.

A. Bringing textbook examples to life

In order to emphasize the point that this learning environment is complementary to instructional texts such as [1-3], examples from these books have been implemented as interactive simulations.



The count-to-infinity problem.

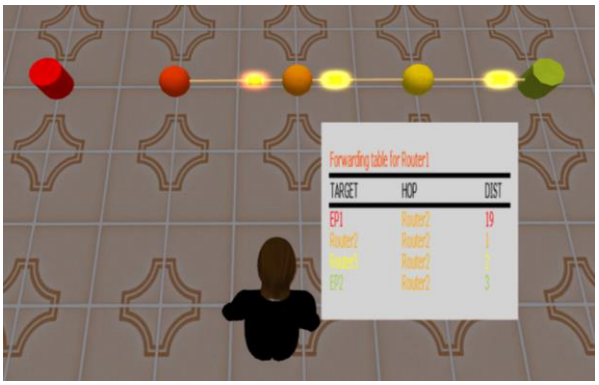


Figure 4: the book diagram (top) and a live simulation of Count to Infinity

Figure 4 shows the "Count to Infinity" example as drawn in [3] and then as rendered in the immersive learning environment. In the simulated version students can control and watch the forwarding tables change for each router as the algorithm runs. This simple network can also be readily built in one of the sandboxes. A student can then break or repair any link while the network is live, thereby seeing a visualization of the algorithm in action. As a further option the algorithm can be changed from DV to OSPF in order to show how a change of protocol can remove that problem.

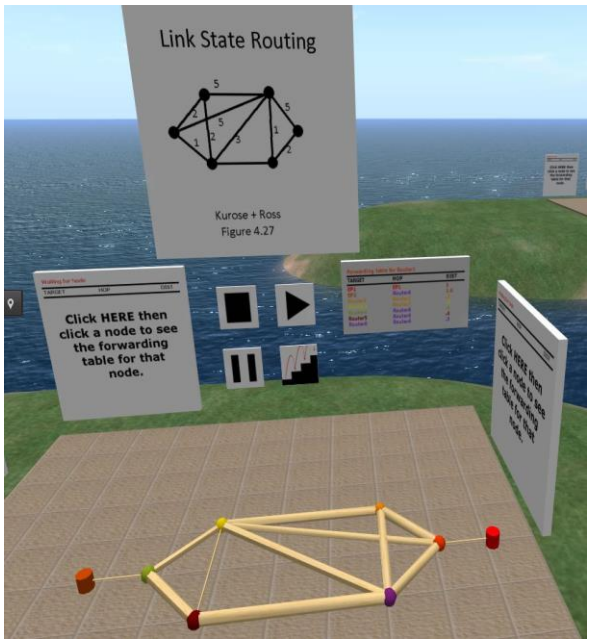


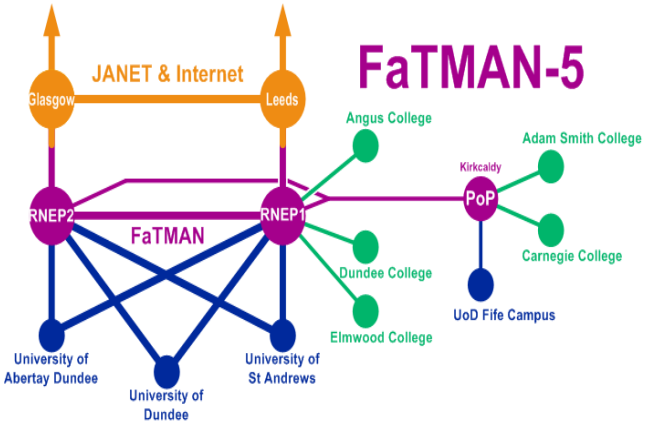
Figure 5: An example from [1] is brought to life

Figure 5 shows a link-state example from [1]. Again, students can study how a router's forwarding tables change as the algorithm runs and examine their final states to check their understanding. Examples from [2] are also provided.

B. *Build your own network*

In order to support exploratory learning learners can build their own network in any of the *shared* or *private* sandbox areas provided. The tools for building a network allow for the creation and deletion of routers, end points and the links between them. The cost of links can be edited – the higher the cost the greater the girth of the visual representation of the connection. Either DV or link-state can be chosen. A live network with packets flowing from one end node to another can have its links broken or new links added in order to see the routing algorithms adapting to the changed configuration.

In a shared sandbox only one user can take control of the building tools but any number of other users can be present and interact using the usual OVW text chat facilities, or even live audio if headsets are used. The private sandbox allows for personal exploration of the tools and network configurations. Topologies can be saved and reloaded for future use.



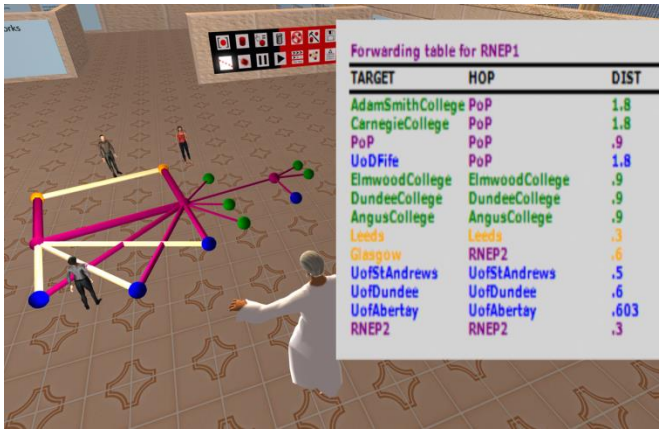


Figure 6: Students model and simulate a ja.net region based on a network diagram; as links are removed and added the forwarding tables at each node can be observed as they dynamically update.

Figure 6 shows a ja.net area, the Fife and Tayside Metropolitan Area Network (FaTMAN), modeled in the simulator by a group of students, with a lecturer on hand to help. Note that the girth of the links between adjacent routers reflects their weight.

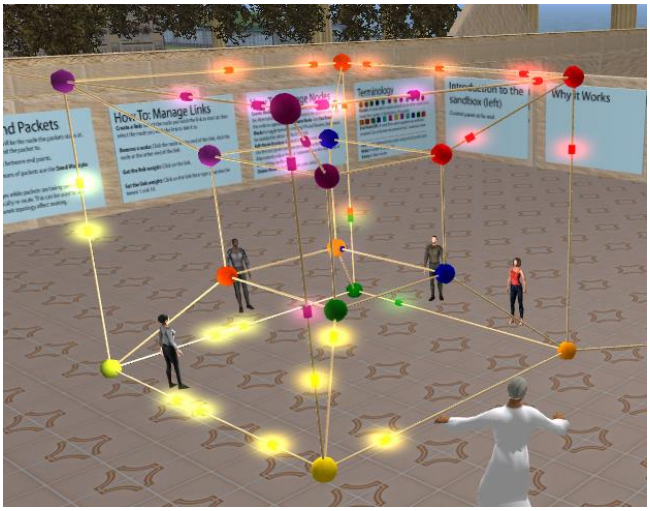


Figure 7: Students collaboratively build a hypercube (k=4) Internet core.

Figure 7 shows how some students have modeled a network routing core as a hypercube of degree 4 rather than the more common fully connected mesh. It is interesting that the 3D immersive environment allows for better modeling and visualization of such structures than the awkward representations such as those shown in Fig. 8.

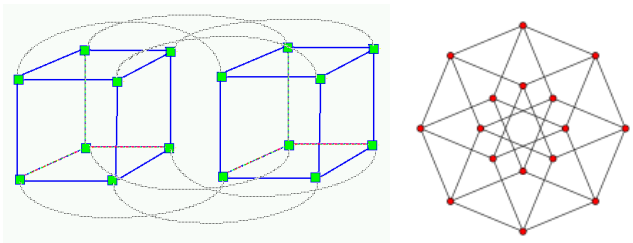


Figure 8: Representing hypercubes (k=4) in 2D

V. How the Routing Simulator Works

A. *Programming Virtual Worlds*

Second Life imposes severe restrictions on coding which makes complex software development extremely difficult. Its native programming language, Linden Scripting Language (LSL) [35], is limited in expressiveness and modularity, code is expected to be written using a basic text editor, and Second Life itself, as a commercial distributed environment, limits the capabilities of code interacting with the host environment in the same way an operating system or hypervisor seeks to protect separate users through resource management and protection.

Routing Island is built on the OpenSim platform and hosted on servers controlled within the university, enabling many more options for programming. OpenSim supports C# as

well as LSL, and as the system is locally controlled the software development capabilities can be extended much further.

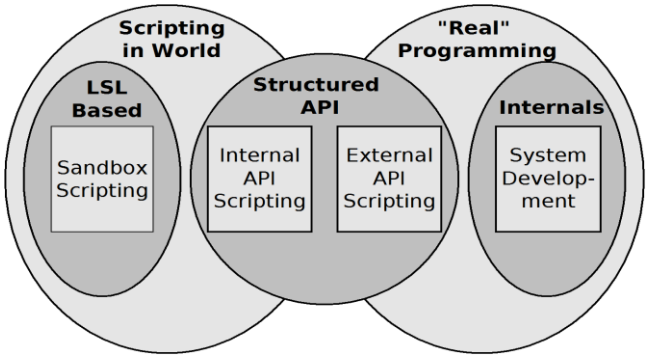


Figure 9: Virtual World programming options

OpenSim provides the following modes for programming content (See Fig. 9):

- Sandbox Scripting: LSL is an example of this, users write LSL scripts with very defined constraints in world. On one level this is powerful, in that it allows in world objects to be programmed to react to events in a rich and varied way but the creation of complex systems is difficult and time consuming.
- Internal API Scripting: An example of this are Mini Region Modules (MRM) and the alternative scripting mechanism they provide. Code is written in C# using an API which gives access to the world's *scenegraph* and also allows external libraries to be loaded in and referenced by its in-world scripts.
- External API development: This is a system whereby a small piece of code, plugged directly into OpenSim then links to an external library which runs as an MRM. The routing simulator was developed an eXtended MRM (XMRM) mechanism that was built to support this functionality. The user writes minimal code in world - all they do is specify a configuration file. This in turn triggers an external library which goes on to load a series of libraries defined in the configuration file. This is the code which links an in-world script to external libraries and so enables the code developed and compiled in an IDE, to run in world.

The XMRM mechanism was implemented in order to create a development environment powerful enough to program the routing simulator.

B. *System Design*

The design of the system is based on the Model, View, Controller pattern. See Fig. 10. Each Module (model, view, control) is defined as an interface in a framework library Each of these interfaces is then implemented in its own library. The entities that make up the system are modelled by another set of interfaces again defined in the framework library. Instances of these interfaces are passed between the modules to share state. Each module then extends the entity interfaces to add the extra functionality it needs.

The various modules are designed to be instantiated in a loosely coupled manner. Because modules only interact through interfaces defined in a separate library different implementations of these interfaces can be swapped out without altering the other modules.

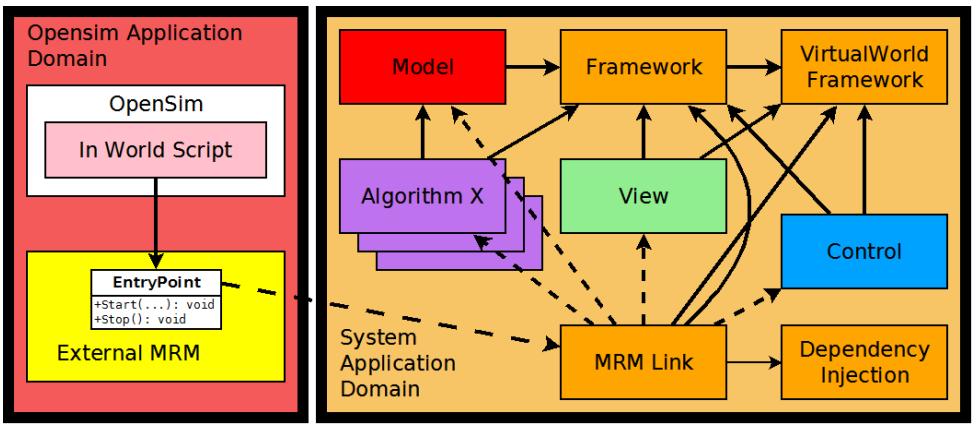


Figure 10: MVC based design for the Routing Island Software Development System

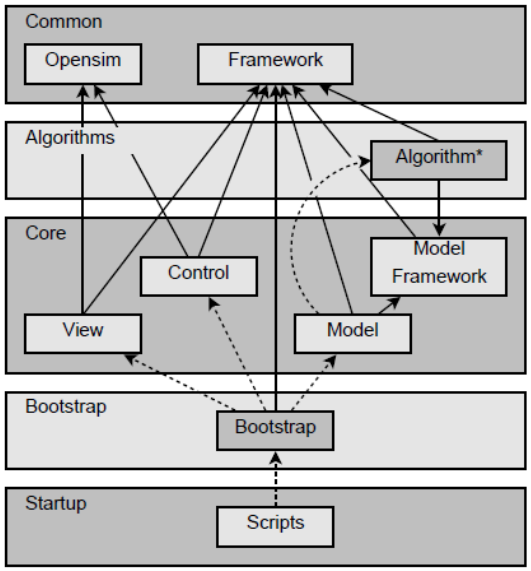


Figure 11: System libraries and their dependencies.

The class which instantiates the modules reads in the locations of the assemblies which implement the interfaces from a config file at run time. This loose coupling means different configurations of the system can be instantiated just by altering the configuration file. All the libraries which make up the system and their dependency relationships are represented in Fig. 11.

Routing Island uses the External MRM system previously outlined to integrate itself into OpenSim (see Fig. 10). Because of the loosely coupled nature of the system any of the modules can be re-implemented or extended to alter their behaviour. There are two mechanisms which are designed specifically to allow simple extensions to be implemented. The first is the algorithm mechanism. The system as a whole is designed to be independent of what routing algorithm is running. The ability to create or alter a topology is completely separate from how the algorithms themselves are implemented.

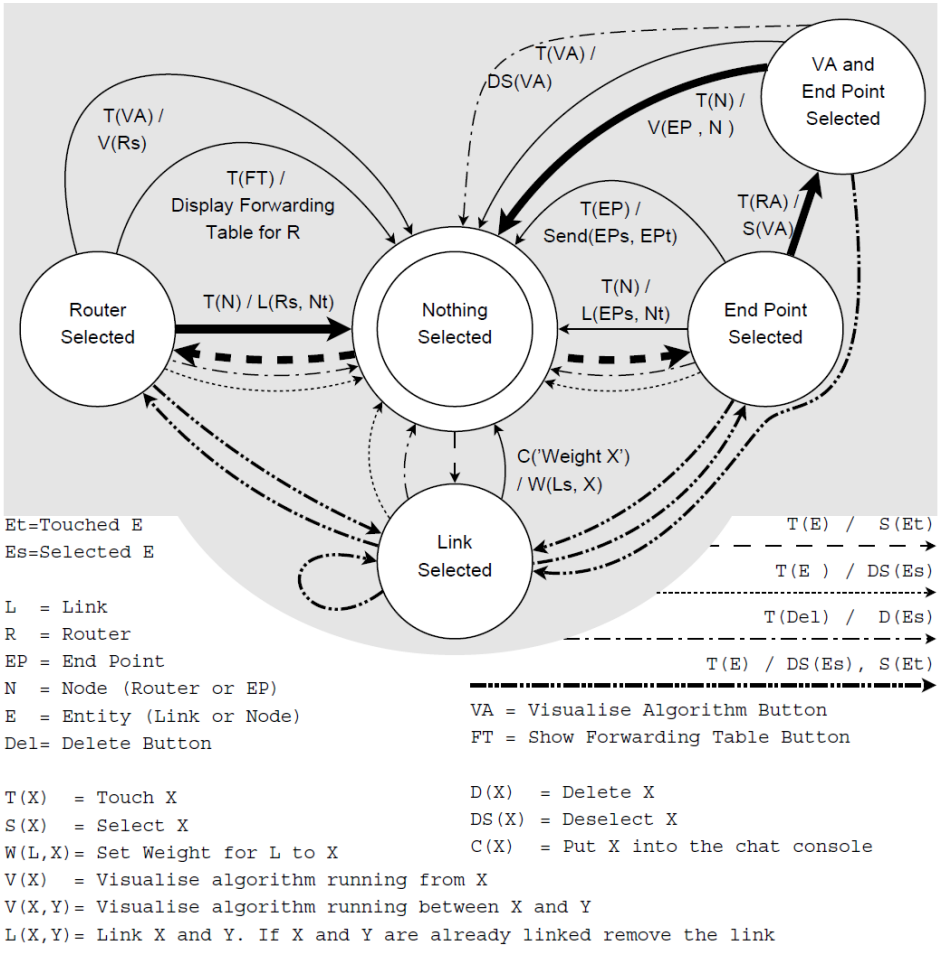


Figure 12: State transition diagram for user interaction

The second extensibility mechanism is how the control module is designed. The control implementations form an inheritance hierarchy. At the top is the Control class which tracks the relationship between all entities (what links link what nodes etc.). This class provides a series of protected utility methods for querying this information and exposes a range of protected events that will be triggered on user input such as a router being touched or a link being deleted. Next the SandboxControl class extends Control. This class exposes no methods at all but registers with the listeners defined in Control and parses this information to produce the Sandbox behaviour described earlier.

Any programmer wishing to alter how the system is interacted with just has to extend one or other of these classes and register with the various listeners to define how the system reacts to different user inputs. The architecture of the system is designed to fully exploit the ability to reference externally compiled libraries from in-world scripts (see Fig 11). The scripts library can be dropped into the main OpenSim directory and referenced by any script. Initialising an instance of an object with the location of a config file hands over control from the in-world script to the externally compiled library.

The instantiated class can then dynamically instantiate a bootstrap class from a library whose location is specified from the configuration file. By changing what configuration file the class is initialised with different systems or configurations can be loaded. The bootstrap class must adhere to an interface defined in the scripts library but the files can be compiled and placed anywhere in the file system. These files can be re-compiled while the system is running in-world and the in-world system can then be restarted and the newly compiled code loaded. This mechanism means that a developer can use an external IDE to develop a system. The developer is able to make changes to the system, re-compile and then type a command in-world to restart the system and see the changes appear. This is a powerful system as the developer is able to utilise all the power of multiple libraries, high level programming and IDE development tools and at the same time behave in-world as if they were just writing and re-loading a script, never having to restart the server.

C. User Interaction

A state transition diagram is shown in Fig. 12. The system is designed to make user interaction as simple as possible. If the user wishes to create a node they simply click on the point on the floor where they wish the node to be. If they wish to link two nodes they click on the first node they wish to link, selecting it, then click on the node they wish to link it to. Similarly if they wish to remove a link they click on the node at one end of the link they wish to remove then click on the node at the other end of the link. State is indicated by the 'glow' effect (see Figs 4 and 7) whereby an object appears to glow. If a node or link is selected it glows, when it ceases to be selected it ceases to glow.

Nodes are subdivided between routers and end points to aid with context sensitivity. Routers can be linked to as many other nodes as is required. End points can only ever be linked to one node. End point nodes allow the user see the system in action. Selecting one source end point and then selecting a destination end point causes a packet to be sent between the two end points.

The way the user interacts with the system is a context based model based on two types of interaction, the notion of a 'touch' and the chat system. A touch is the basic method of interaction in OpenSim. If an in-world primitive has a listener attached then whenever a user clicks on that primitive a touch event is triggered.

The input system works with two separate classes of objects. Firstly there are control primitives. These are labelled and provide specific functions. Secondly there are the primitives which make up the simulation, known as entities. Control prims are further divided into three groups, buttons, toggles and floor. Buttons cause actions to happen, toggles set state for a specific parameter and floor can be clicked on to create entities in-world. Entities are divided between links, end points, routers and packets. Packets are not interacted with directly, the other three forms have specific properties. The notion of control primitives seeks to give users extra control of the system.

Toggles allow users to set system state e.g. between sending one packet at a time or sending a stream of packets. Buttons allow the user to run specific tasks over the topology they have defined, e.g. running a routing algorithm or speeding up the operation of the system.

More specific control is gained using the chat mechanism. Using commands chatted into the global chat window the user can perform tasks such as saving the topology they have created to an XML file or changing the weight of a link.

How the user actually experiences the system is through nodes and links defining a topology. The system can be configured to load in a static topology from a configuration file. This can be set up so the user cannot alter the topology but can run routing algorithms over the topology. These pre-defined topologies serve a similar purpose to animating algorithms, the difference being the algorithm is actually running dynamically in the simulator. Thus the topology can be changed and the algorithm will change accordingly. These topologies can be used to explain specific cases such as the examples from textbooks described earlier or provide a detailed textual description explaining the precise topology being displayed. Of course, the distinguishing feature of the system is that the user has the power to create and link nodes in sandbox mode.

VI. Conclusion

This paper has described the development and deployment of Routing Archipelago, a learning aid for Internet Routing algorithms developed within an Open Virtual World. The development was motivated by the need for exploratory learning facilities to complement textbooks such as [1-3] and online courses such as [4], and by the observation that learners readily and enthusiastically engage with virtual worlds [8, 9].

The archipelago is located within a multi-user 3D model accessible via the Internet. It acts as an interface to a large range of learning resources, including web pages, videos, standards documents and animations. Learners can choose their own pathways to navigate these resources and lecturers can use it for demonstration purposes. Collaborative learning strategies are encouraged as the actions and navigation of learners is achieved through their avatars. The creation of a new software engineering approach featuring XMRMs was necessitated by the complexity of this project. It has unlocked the potential of Open Virtual Worlds transforming OpenSim into a rapid application development for 3D applications and enabling powerful new modes of interaction. In the case of Internet routing it allows for the creation of integrated and visual 3D simulations of network routing protocols. Educationalists can define scenarios which learners can step through, observing the interactions between network topology, routing algorithms and traffic flows. In additions learners may individually or collaboratively define their own network topologies, data flows and algorithms and step through simulations. Integration within the learning environment means that it is possible to change the topology of a simulation whilst it is live. Users can thus investigate the effectiveness of routing protocols in managing topological changes. These systems have been used on multiple degree levels within and credit bearing courses within HE. Students have found them to be engaging, stimulating and intuitive to use.

References

[1] J. Kurose and K. Ross, *Computer Networking: A Top-Down Approach, 6th ed.*, 6 ed.: Pearson, 2012.

[2] L. Peterson and B. Davie, *Computer Networks: A Systems Approach (5th ed.)*, 5 ed.: Morgan Kaufmann, 2011.

[3] A. S. Tanenbaum and D. Wetherall, *Computer Networks*, 5th ed.: Pearson, 2010.

[4] P. Levis and N. McKeown. (2014, April 2014). *An Introduction to Computer Networks; https://class.stanford.edu/courses/Engineering/Networking/Winter2014/about*. Available: https://class.stanford.edu/courses/Engineering/Networking/Winter2014/about

[5] T. Bates. (2012, April 2014). What's right and what's wrong about Coursera-style MOOCs; http://www.tonybates.ca/2012/08/05/whats-right-and-whats-wrong-about-coursera-style-moocs/.

[6] L. W. Anderson, D. R. Krathwohl, P. W. Airasian, K. A. Cruikshank, R. E. Mayer, P. R. Pintrich, J. Raths, and M. C. Wittrock, *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives*. New York, NY: Longman, 2001.

[7] U. Fuller, C. G. Johnson, T. Ahoniemi, D. Cukierman, I. Hern, #225, n-Losada, J. Jackova, E. Lahtinen, T. L. Lewis, D. M. Thompson, C. Riedesel, and E. Thompson, "Developing a computer science-specific learning taxonomy," presented at the Working group reports on ACM ITiCSE on Innovation and technology in computer science education, Dundee, Scotland, 2007.

[8] P. Sancho, J. Tirrente, and B. Fernandez-Manjon, "Fighting Drop-Out Rates in Engineering Education: Empirical Research regarding the Impact of MUVEs on Students' Motivation," presented at the 1st European Imeersive Education Summit, Madrid, 2011.

[9] G. I. U. S. Perera, C. Allison, T. Sturgeon, and J. R. Nicoll, "Towards Successful 3D Virtual Learning - A Case Study on Teaching Human Computer Interaction," in *the 4th International Conference for Internet Technology and Secured Transactions*, London, 2009.

[10] T. Sturgeon, C. Allison, and A. Miller, "802.11 Wireless Experiments in a Virtual World," *ACM SIGCSE Bull.*, vol. 41, pp. 85-89, 2009 2009.

[11] L. D. Thomas and P. Mead, " Implementation of Second Life in electromagnetic theory course; http://www.youtube.com/watch?v=ExWX4Kz-6Qc," presented at the 38th Frontiers in Education Conference, 2008. , Saratoga Springs, NY, 2008.

[12] M. Esteves, B. Fonseca, and L. Morgado, "Using Second Life for Problem Based Learning in computer science programming," *Journal of Virtual Worlds Research*, vol. 2, 2009.

[13] R. S. Clavering and A. R. Nicols, "Lessons learned implementing an educational system in Second Life," presented at the Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it, University of Lancaster, United Kingdom, 2007.

[14] K. Getchell, A. Miller, R. Nicoll, R. Sweetman, and C. Allison, "Games Methodologies and Immersive Environments for Virtual Fieldwork," *IEEE Transactions on Learning Technologies*, vol. 3, pp. 281-293, 2010.

[15] NASA. (2010, Learning Technologies Project FY10 Performance Report Project. http://www.nasa.gov/pdf/505587main_2010_ESE_LT.pdf.

[16] F. Bellotti, R. Berta, A. Gloria, and L. Primavera, "Designing Online Virtual Worlds for Cultural Heritage," in *Information and Communication Technologies in Tourism 2009*, W. Höpken, U. Gretzel, and R. Law, Eds., ed: Springer Vienna, 2009, pp. 199-209.

[17] L. Dow, A. Miller, and E. Burt, "Managing Humanitarian Emergencies: Teaching and Learning with a Virtual Humanitarian Disaster Tool," in *4th International Conference on Computers in Education, CSEDU 2012*, Porftugal, 2012.

[18] W. L. Johnson, "Developing Intercultural Competence through Videogames, Keynote speech," in *ACM International Workshop on Intercultural Collaboration (IWIC) 2009*, Stanford Unviersity, Palo Alto, California, 2009, pp. 99-100.

[19] Linden_Labs. (2003, October). *Second Life, www.secondlife.com*.

[20] S. Warburton, "Second Life in Higher Education: assessing the potential for and the barriers to deploying virtual worlds in learning and teaching," *British Journal of Educational Technology,* vol. 40, pp. 414-426, 2009.

[21] C. Allison, T. Sturgeon, A. Miller, G. I. U. S. Perera, and J. R. N. Nicoll, "Educationally Enhanced Virtual Worlds," presented at the 40th IEEE Frontiers in Education Conference, 2010. FIE '10., Washington, 2010.

[22] The_Open_Simulator_Project. (2010, April). *OpenSim: http://opensimulator.org/wiki/Main_Page*.

[23] M. Janus, "hippo: http://sourceforge.net/projects/opensim-viewer/," ed, 2010.

[24] unknown, "meerkat: http://code.google.com/p/meerkat-viewer/downloads/list," ed, 2009.

[25] P. V. P. Inc., "Phoenix: http://www.phoenixviewer.com/," ed, 2012.

[26] Y. Rekhter, T. Li, and S. Hares, "Request for Comments: 4271; Border Gateway Protocol 4; http://tools.ietf.org/html/rfc4271," IETF2006.

[27] C. Hedrick, "RIP: Routing Information Protocol, http://tools.ietf.org/html/rfc1058, RFC 1058," IETF1988.

[28] G. Malkin, "RIP Version 2, http://tools.ietf.org/html/rfc2453, RFC 2453," IETF1998.

[29] J. Moy, "OSPF Version 2, http://www.ietf.org/rfc/rfc2328.txt, RFC 2328," IETF1998.

[30] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik,* vol. 1, pp. 269-271, 1959.

[31] Wikipedia, "Internet Routing http://en.wikipedia.org/wiki/Routing_protocol," ed. Wikipedia, 2014.

[32] VambarInc. (2012, October 2012). Distance Vector and Link State Protocols, http://www.youtube.com/watch?v=ygxBBMztT4U&feature=relmfu.

[33] C. Allison and B. Ling, "OSPF Routing Online Tutorial; http://ospf.cs.st-andrews.ac.uk," 2005.

[34] C. Laffra, "Dijkstra's Shortest Path Algorithm, http://www.dgp.toronto.edu/people/JamesStewart/270/9798s/Laffra/DijkstraApplet.html," ed, 2005.

[35] Linden_Labs. (2013, October). *Linden Scripting Language; http://wiki.secondlife.com/wiki/LSL_Portal*.